INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®

Assistant Assignment Planner System Design and Implementation

JUN LIU

A Major Report

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

March 2002

© Jun Liu 2002

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-68473-3

Canada

Abstract

Assistant Assignment Planner System

Design and Implementation

Jun Liu

This report describes the system design and implementation for the Teaching Assistant Assignment Planner System (TAAP) for Computer Science Department in Concordia University. TAAP system is expected to provide a convenient way for TA administrator to manage TA assignment task. This report also presents an Object-Oriented approach for the implementation of Graphical User Interface (GUI) for TAAP system in Windows environment. UML (Unified Modeling Language) has been used in the design and analysis procedure of TAAP, and we use Borland Jbuilder 5.0 for the implementation of the user interface. Microsoft Access 2000 has been used to implement the database of the system.

Acknowledgement

I would like to thank my supervisor, Dr. Peter Grogono for his patience and valuable suggestions.

Helpful discussions were had with other members of the TAAP project, especially with Jiantao He and Xiang Wang.

Table of Contents

List of Figures

CHAPTER 1

INTRODUCTION

This major report, which is under the supervision of Prof. Peter Grogono, is originated from an actual requirement in the Computer Science Department of Concordia University.

Every year staffs in Computer Science Department need to spend a lot of time in processing "Teaching Assistant" (TA) applications before each new term. There normally are more than a hundred of TA tasks posted every term with dozens of applicants on the list. Therefore, the total number of applications can easily top hundreds. The information of tasks, applicants and corresponding applications are stored in the database located in a UNIX machine. The basic process of doing TA assignments is to first pick up an unassigned task from the task table, then to check the application table to get the information of the applicant(s) who have applied this task, and finally reflect the new assignment in the application table. In order to obtain detailed information about an individual applicant, the applicant table has to be consulted constantly during this process.

So far the whole assignment process is still carried on manually by a TA Coordinator, and no tools have been developed for them to help facilitate this routine and mandatory task. Assigning TAs is a tedious and error-prone process which means not only time consuming, but also likely susceptible to such errors as assignment schedule conflicts (i.e. one applicant gets two tasks with conflict schedule), multiple assignments (i.e. the same task is assigned to more than one applicant) and overloaded assignments. (i.e. some applicants are too overloaded).

Without a tool, it is also hard for the TA coordinators to get the progressing information about how many and what tasks have been assigned and how many and what tasks are yet to be assigned.

"Teaching Assistant Assignment Planner" (TAAP) demonstrated in this major report is the tool intended to help automate the assignment process in the manner of eliminating human errors, minimizing the memorization request and providing the progressing information and status report. The goal is to try to keep the system itself as simple as possible, and the productivity is obtained by focusing on features like user-friendly interface, easy-to-use and easy-to-maintain.

1.1 TAAP SYSTEM

TAAP, as mentioned before, is a semi-automation tool to help simplify the TA assignment process. However, the actual assignment still needs to be done manually. TAAP guarantees the better productivity by providing functionalities to assist TA assigners doing their job not only effectively and efficiently, but also correctly. The automatic assignment mechanism (which usually requires a sophisticated algorithm) is not considered here, thus not implemented in TAAP. It might be part of the TAAP implementation for the next phase.

The main functionalities implemented in TAAP are categorized as follows:

- Assigning tasks and deleting assigned tasks
- Viewing TAs, assigned Tasks and unassigned Tasks.
- Reporting TAs, Tasks and Assignments.

TAAP is a local integrated database application system with both client-side application and back-end database running on the same machine, therefore it can only be

accessed from where TAAP is installed, any attempt for remote access is not possible. TAAP is developed and operated solely on the Windows platform (9x/2000/NT). For the sake of clarity, TAAP will be referred as the client-side application hereafter, while back-end database will be referred using the database name.

1.2 Organization of This Report

Chapter 1 presents the introduction of this report. Chapter 2 presents the background on of TAAP system. Chapter 3 presents the object-oriented design for TAAP system. Chapter 4 describes the implementation of TAAP system. Chapter 5 presents the Result. A brief summary of the TAAP system, the contribution of this report, and the suggested future work is given in Chapter 6.

We assume the readers are familiar with Object-Oriented modeling in Unified Modeling Language (UML).

CHAPTER 2

BACKGROUND

This chapter describes the background knowledge necessary for designing and implementing the TAAP system

2.1 ACRONYMS AND DEFINITIONS

- **AWT**

  Abstract Window Toolkit, it is a large collection of classes for building graphical user interfaces in Java.

- **Jtable**

  Jtable is a Java Swing component.

- **JcomboBox**

  A GUI independent object ComboBox in ComBox.java class. This object belongs to the java.awt package.

- **GUI**

  Graphic User Interface. A GUI is what computer types call the system of icons, toolbars and other objects that our computer use to display and access information.

- **SQL**

  Structured Query Language. SQL is a kind of query language in which a user requests information from the database. Query Languages are typically of level higher than that of a standard programming language.

- **Swing**

  Swing refers to the new library of GUI controls (buttons, sliders, checkboxes, etc.) that replaces some weak and inflexible AWT controls.

- **TA**

  Teaching Assistant. In TA table, A TA has some fixed characteristics and some data that changes as user makes assignments. The fixed (or given) characteristics are:

  Name A string of characters.

  Experience A list of the tasks that the TA can perform.

  Tasks A list of the tasks actually performed by the TA.

- **Tasks**

  Each course is offered in zero or more sections. A Task is described by the following data:

  Course A four-letter string: COMP, ENCS.

  Number A three- or four-digit string: 248, 5421.

  Session A single digit indicating the term: 1, 2, 4.

  Title The name of the course. Artificial Intelligence.

  Section The code for a section of the course: AA

  Task type A task associated with this section: TUT, LAB, MRK.

  Days Days on which the tutorial/lab is offered: T, W, F.

  Start The start time of the tutorial/lab: 10:15.

  Finish The finishing time of the tutorial/lab: 11:30.

- **Task Flow Diagram**

Task flow analysis will document the details of specific tasks. It can include details of interactions between the user and the current system, or other individuals, and any problems related to them. Copies of screens from the current system may also be taken to provide details of interactive tasks. Task flows will not only show the specific details of current work processes but may also highlight areas where task processes are poorly understood, are carried out differently by different staff, or are inconsistent with the higher level task structure.

- **Usability**

  Usability is a high-level quality objective: The extent to which a product (UI) with effectiveness, efficiency and satisfaction in a specified context of use. [ISO9241-11]

2.2 TAAP GUI DESIGN PRINCIPLES

Easy-to-use and Easy-to-maintain are our goals to design TAAP system. Graphical User Interfaces (GUI) have been used for many years, and most users interact with computer through GUI now. Due to 50% of interactive systems development efforts are related to user interface, User Interface should be considered as an important component for the organization and development of the overall interactive system.

Since TAAP system is a local integrated database application system with both client-side application and back-end database running on the same machine, the interface of the TAAP system design uses the following guidelines:

- Understandability

The extent bears on measuring the difficulty of user on understanding software functions, operations and concepts while user has no previous knowledge about software.

The TAAP system just has one main window for user to do their task, so it is easy to learn. Simplicity is important feature for user to understand the system easily.

- Operability

  Measures the user's effort for operation and operation control.

  In order to prevent user from becoming lost, TAAP system seldom opens a new window, users could accomplish their goals in one window.

- Efficiency

  The amount of resources expended in relation to the accuracy and completeness with which user achieves a goal.

  The TAAP system interface tries to minimize the steps that users have to follow to achieve their goals.

- Completeness

  The extent to which the user can complete a specified task. The TAAP system has been designed to meet the user requirements, and all the functionalities are implemented.

- Flexibility

  Indicates the degree of possible modification to user interface by the user. User Interface should be easy to update when requirement changes.

  The above quality-in-use factors are also defined in [ISO-9126].

2.3 HARDWARE ENVIRONMENT

Because TAAP system is a local database application system, one personal computer is required for system installation and operation. The requirements for the personal computer are at least 266M processor, 128M RAM and 50M hard disk free storage space [5].

These requirements are suggested by Borland Company for installing and running Jbuilder 5.0 software, which is used for TAAP system implementation.

2.4 SOFTWARE ENVIRONMENT

Since Windows 95/98 has weak security feature, Windows NT operating system is recommended for the TAAP system. Borland JBuilder 5.0 is chosen for the implementation of TAAP system because JBuilder supports visually designing and programming Java classes. However, considering the portability of TAAP system, the Jbuilder specific Java features could not be used, instead of standard AWT/SWING tool kits have been used. Microsoft Access database is chosen because Access fulfills current requirements for TAAP system.

CHAPTER 3

SYSTEM DESIGN

3.1 TAAP SYSTEM REQUIREMENTS ANALYSIS

3.1.1 USE CASE DIAGRAM

The Use Case diagram for TAAP system lists below, which illustrates an overview of Use Cases for end-user (lab coordinator). This Use Case Diagram is also used to precisely describe the functional requirements of TAAP system [1].

We define only one actor Lab coordinator in TAAP system. According to the requirements of TAAP system, we divide use Cases into 5 parts, which are Assign TA, View TA, View Task, View Assignment, and Help use case. Each task in turn may have several sub-usecases.

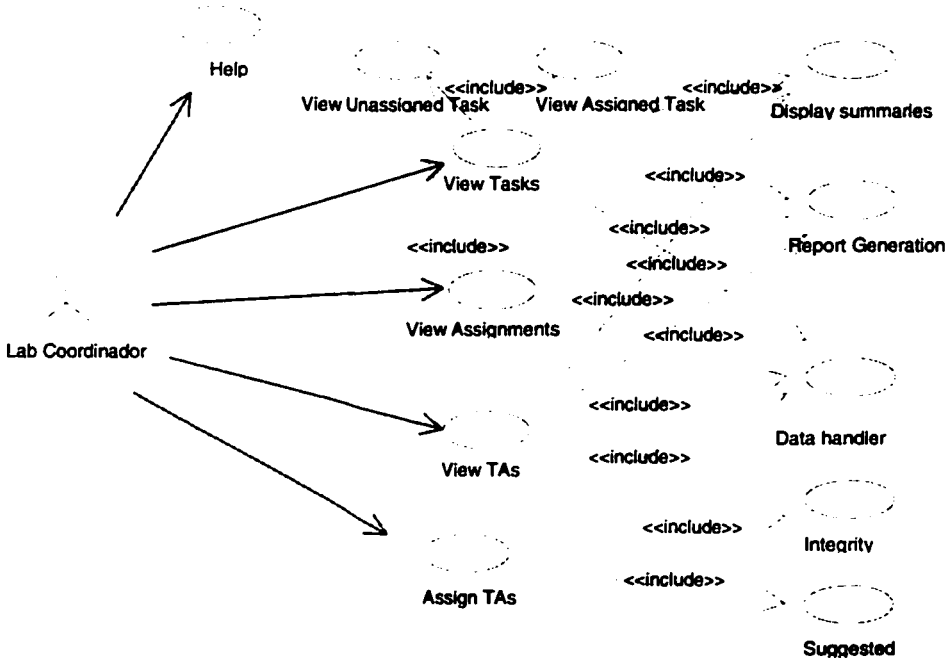

Figure 3-1: Use Case Diagram for TAAP system

- View TAs

The program displays all TAs, together with TA's Name, Student ID, Applied Courses, Task Type and Assign Status. This enables the user to check progress and to see which TAs are still short of work.

- View Tasks

The program displays all tasks. Each task is listed with the TA's name if a TA has been assigned, this field is blank if no TA has been assigned.

- View Assigned Tasks

The program displays only the assigned tasks

- View Unassigned Tasks

The program displays only the unassigned tasks

- Report Generation

The program generates a report.

- Summary

The program provides summary data. Program will display the summary data on the screen at all times.

(a) Number of TAs.

(b) Number of tasks.

(c) Number of tasks with a TA assigned.

- Data Handler

Data Handler is for handling data and accessing the database.

- Suggest TAs

The program is displaying unassigned tasks. If the user selects a task, the program suggests suitable TAs (that is, the TAs whose Experience field includes this task).

- Assign TAs

The user selects a TA and a task; the program assigns the task to the TA and checks for conflicts. The program is displaying a section and a list of suggested TAs, clicking on a TA assigns the TA to that task.

- Integrity Check

When assign a Task to a TA, the program checks the time to make sure that the time of assigned tasks will not conflict. (One TA should not be assigned more than one task at a certain time).

- Help

The program provides help information for user to use the TAAP system.

3.1.2 TASK ANALYSIS

There are four different major tasks in TAAP system (Assign TA task is other team member's responsibility, it will not be discussed in this section): View TAs, View Tasks, View Assignment and Generate Report. In generate report task part, we present two examples which are Generate TA Report sub-task and Generate Assignment Report sub-task. The detailed descriptions for the tasks are given as below:

- View TAs: The Task starts when user selects View TAs function. The program could display all TAs, or the TAs in specified department upon request, together with their fixed and variable data, such as Name, SID, Applied_Course, Task_Type and Assign_Status. This enables the user to

check progress and to see which TAs are still short of work. The user can also generate a report file and save to disk. The task flow diagram is shown in Figure 3-2.

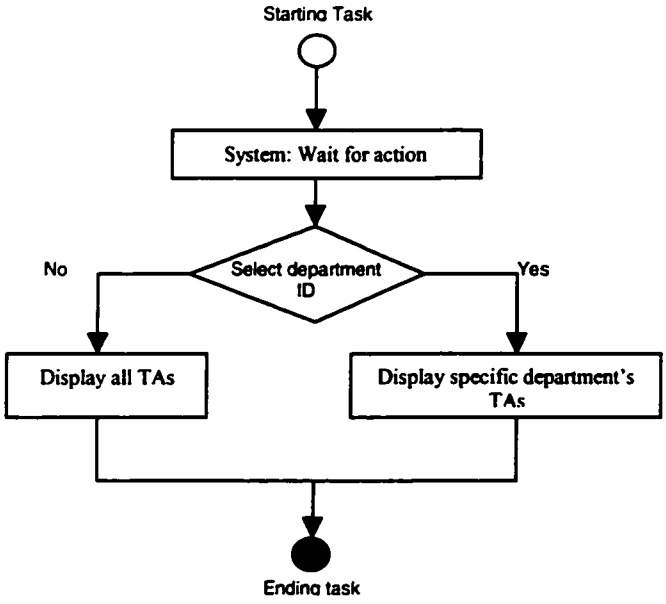

Figure 3-2: View TAs Task Flow Diagram

- View Tasks: This Task starts when user selects View Tasks function. The program could display all tasks, or display tasks in specific department and task type on request. Each task is listed with the TA's name if a TA has been assigned; this field is blank if no TA has been assigned. If no one assigned for a specific task, the TAs who applied for this task will be listed.

  The user can also generate a report file. The task flow diagram is shown in Figure 3-3.

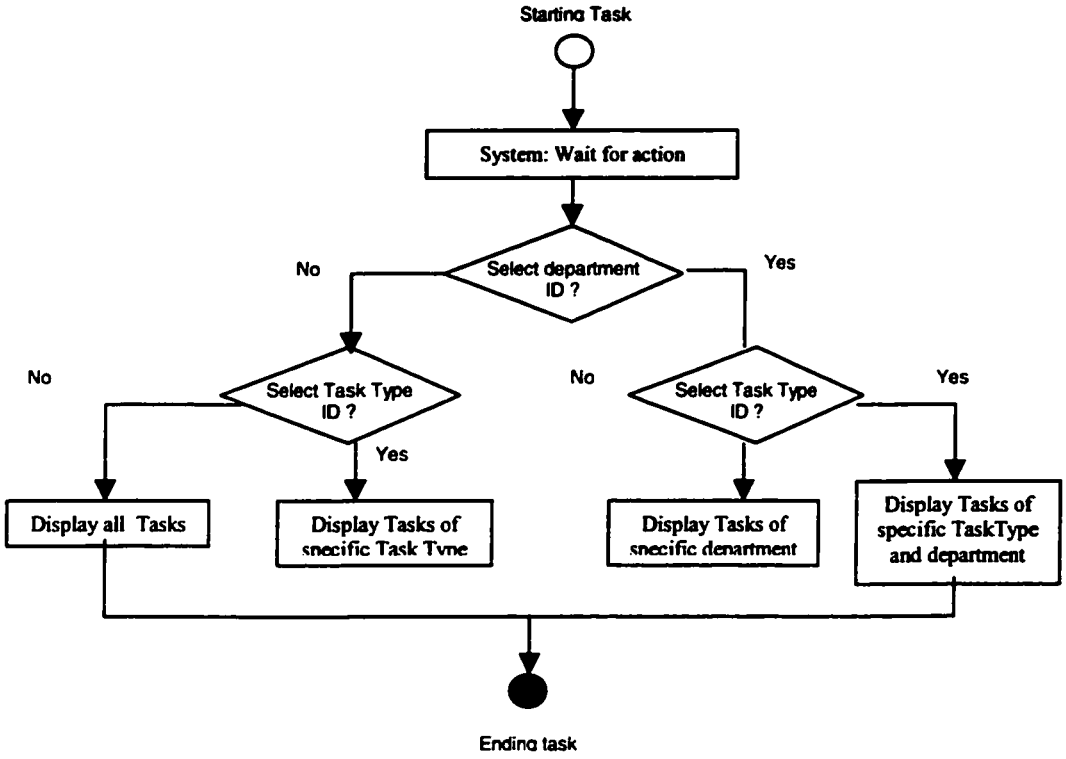

Figure 3-3: View Tasks Task Flow Diagram

- View Assignments: This Task starts when user selects View Assignments function. The program display all assigned tasks sorted by task ID, session and course on request. The unassigned tasks can also be displayed on request. Program can also generate a report file. The task flow diagram is shown in Figure 3-4.

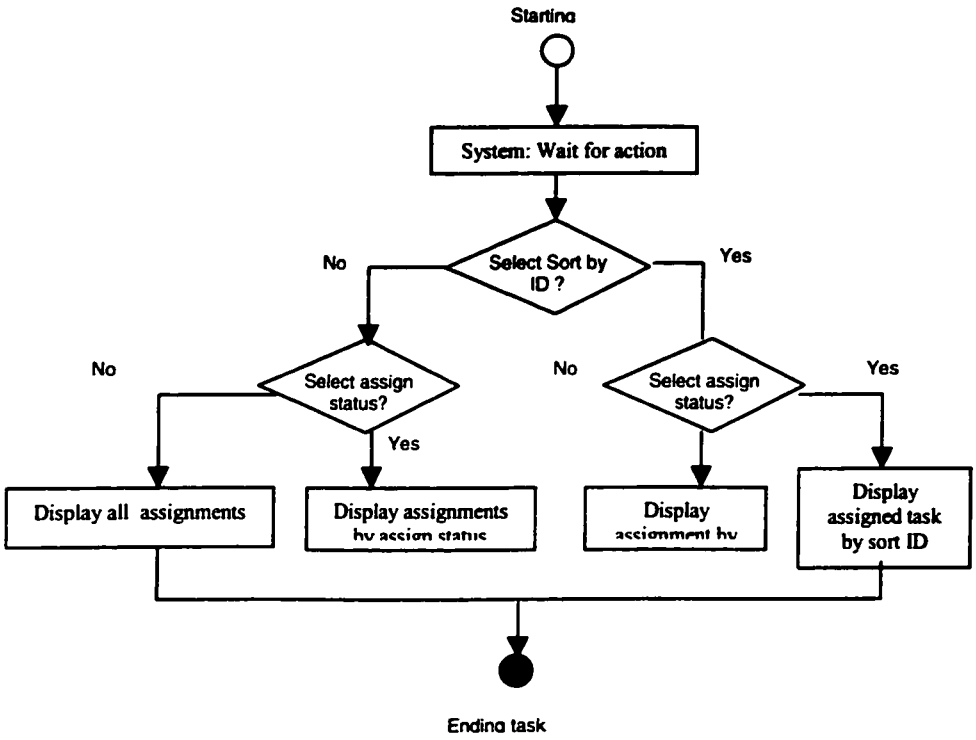

Figure 3-4: View Assignments Task Flow Diagram

- Generate Assignment Report: This sub-task starts when users finish View Assignments task The report can also be displayed / saved on request. The task flow diagram is shown in Figure 3-5.

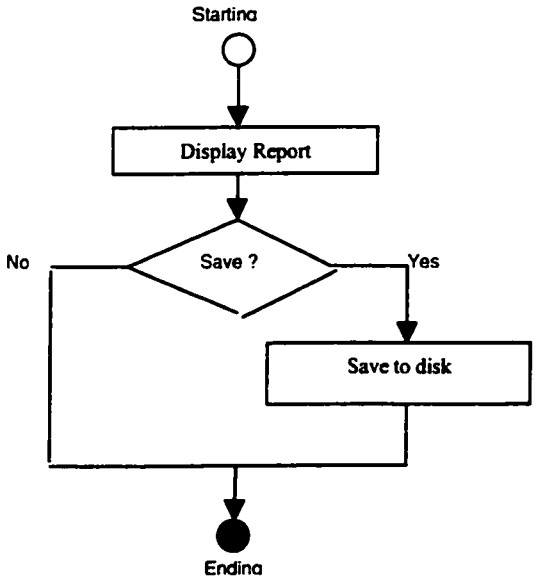

Figure 3-5: Generate View Assignments Report Task Flow Diagram

- Generate TA Report: This sub-task starts when users finish View TAs task. The report can also be displayed / saved on request. The task flow diagram is shown in Figure 3-6.

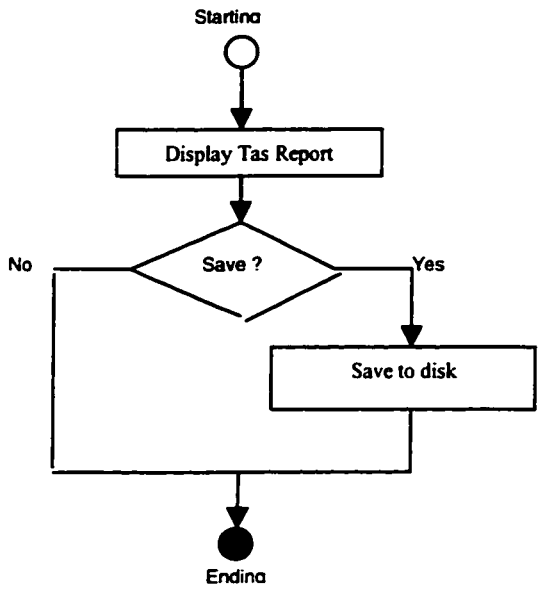

Figure 3-6: Generate View TAs Report Task Flow Diagram

3.2 SYSTEM ARCHITECTURE

System architecture design is the high-level strategy for solving the target problem and building a solution. We use the layered architecture to design TAAP system. There are four layers in the TAAP system architecture.

- Presentation layer

  Contains the ViewTA, ViewTask ViewAssignment and AssignTA objects which comprise the GUI of the TAAP system.

- Domain Logic

  There is only one class (Report) designed for Application Logic layer, we also encapsulate the domain-specific process such as DisplayTAs, GenerateReport

methods into all GUI classes such ViewTA. Logically, the domain logic is an independent layer in TAAP system.

- Persistence

  Contains the DataInterface object to provide Read/Write access to the database. Concretely, DataInterface is implemented as a Java class that supports Create, Read, Update and Delete operations through a set of interfaces.

- Data Storage

  Contains the mechanism employed for storing data persistently. Access database is used for data storage in TAAP system.

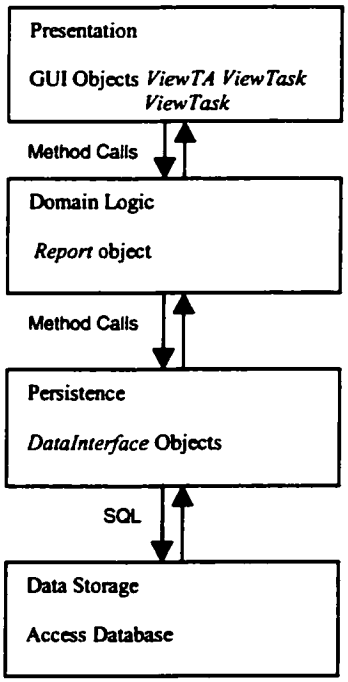

Figure 3-7: TAAP System Layered Architecture Diagram

A user interacts with TAAP system through GUI part. All requests from user send to DataInterface object to produce the standard SQL command and pass the query requests to Database. The result of an SQL query will send back to GUI module.

DataInterface class provides data service, it directly accesses the database through JDBC driver. All the database operations and queries encapsulated in class within DataInterface class.

3.3 CLASS DIAGRAM

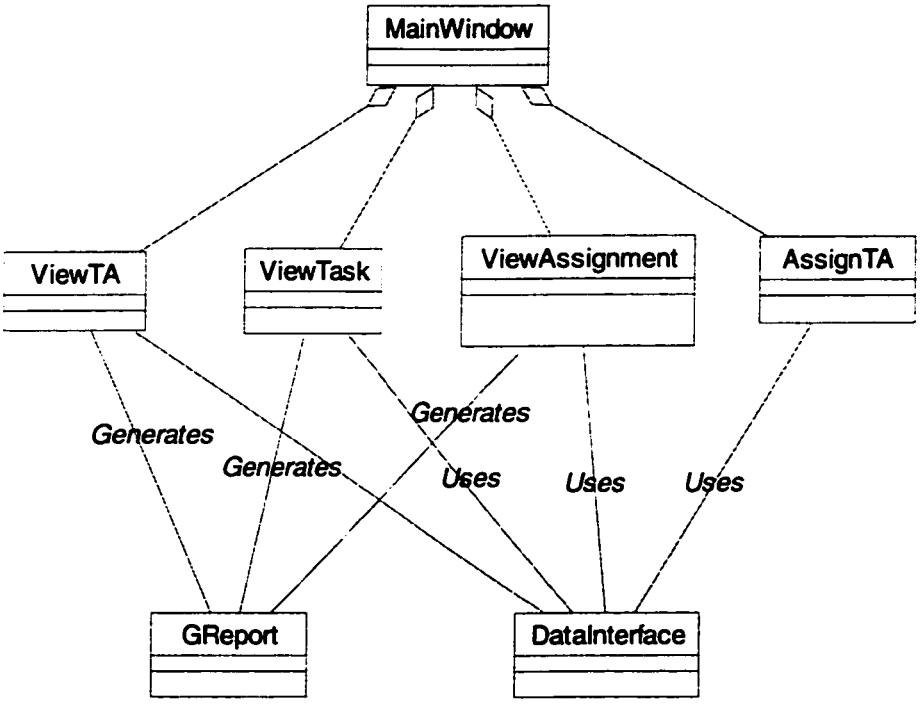

Figure 3-8: TAAP System Class Diagram

The class diagram for TAAP system is presented in Figure 3-8. It describes the types of objects in TAAP system and the relationship existed among them.

When TAAP system is started, MainWindow graphic object will be created. Upon the run-time situation, it will create ViewTA graphic object if user click on "View TA"

button in main window. The same way, VeiwTask, ViewAssignment or AssignTA object will be created if the corresponding button is clicked.

All TAs, Tasks and Assignments data request to database will be send to Database Interface module. When DataInterface receives data request from ViewTA, ViewTask, or ViewAssignment, it will get the data records from Access database and return them to the caller.

Report object will be generated when a request to report is received by ViewTA, ViewTask, or ViewAssignment object. Report object will save report data to file on request.

3.4 SEQUENCE DIAGRAM

We present following sequence diagrams for the major use cases in TAAP system.

3.4.1 View TA Use Case

The system sequence diagram for View TA Use Case is presented in Figure 3-9. The use case starts from user clicking on "ViewTA" button. MainWindow in turn will call (deck.getLayout()).show() to show ViewTA panel. User then selects Course in which TAs registered. Step by step, user enters display command by clicking on "Display" button, the data which input by user then send to DataInterface by calling listTA() function. DataInterface will get query result from database and return to ViewTA. The data finally will be displayed in MainWindow for user.

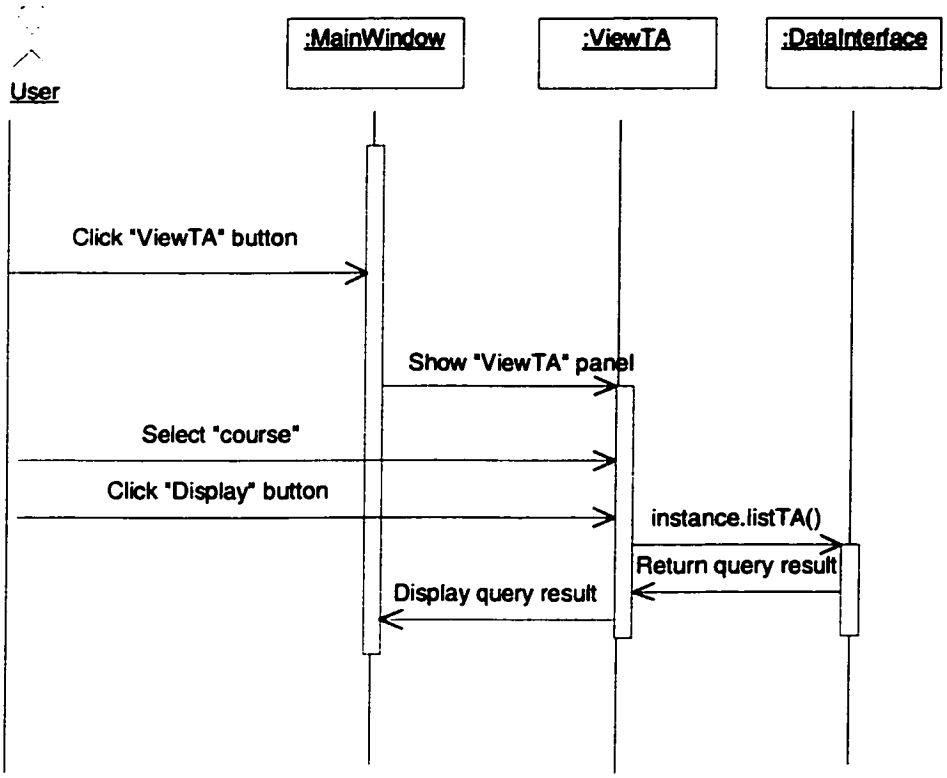

Figure 3-9: System Sequence Diagram: View TA

3.4.2 View Task Use Case

The system sequence diagram for View Task Use Case is presented in Figure 3-10. The use case starts from user clicking on "ViewTask" button. MainWindow in turn will call (deck.getLayout()).show() to show ViewTask panel. User then select Course in which TAs registered, and select Task Type which TAs need to do. Step by step, user enters display command by clicking on "Display" button, the data which input by user then send to DataInterface by calling listAllTask() function. DataInterface will get query result from database and return to ViewTask. The data finally will be displayed in MainWindow for user.

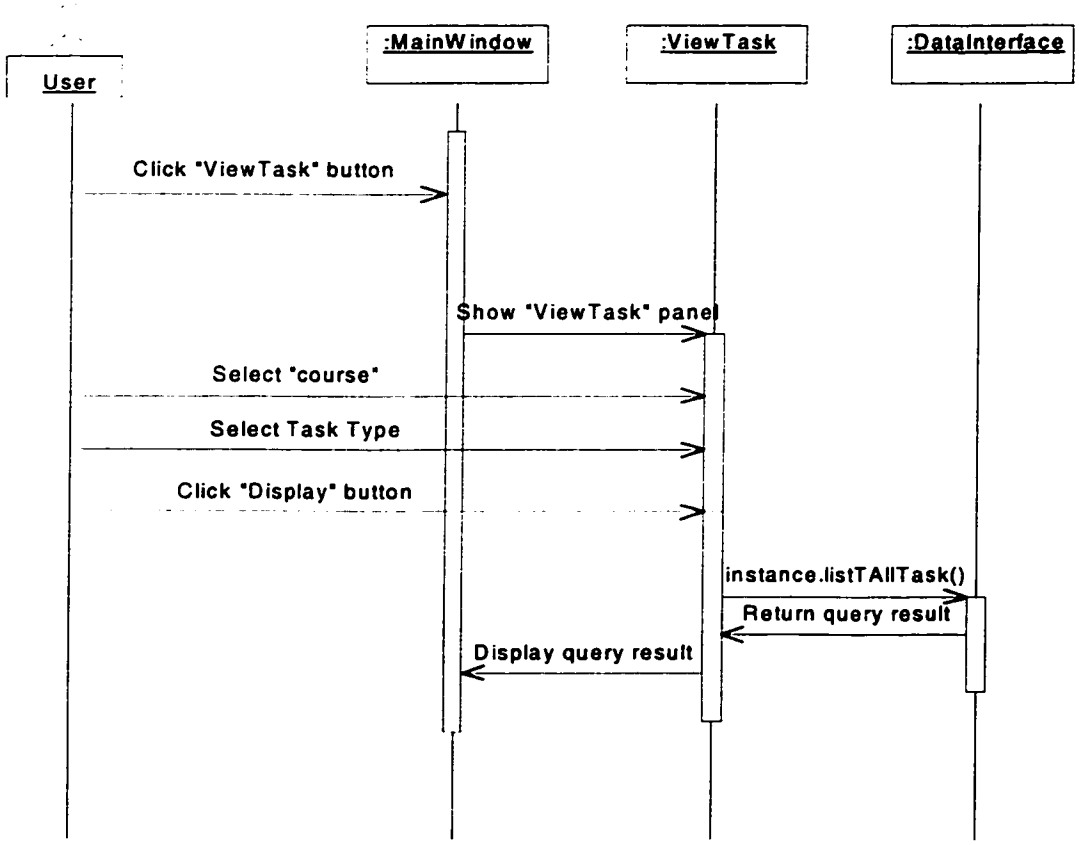

Figure 3-10: System Sequence Diagram: View Task

3.4.3 View Unassigned Task Use Case

The system sequence diagram for View Unassigned Task Use Case is presented in Figure 3-11. The use case starts from user clicking on "ViewTask" button. MainWindow in turn will call (deck.getLayout()).show() to show ViewTask panel. User then selects Course in which TAs registered, Task Type which TAs need to do. Unassigned Task CheckBox in turn is selected. Step by step, user enters display command by clicking on "Display" button, the data which input by user then send to DataInterface by calling listTUnassignedTask() function. DataInterface will get query result from database and return to ViewTask. The data finally will be displayed in MainWindow for user.

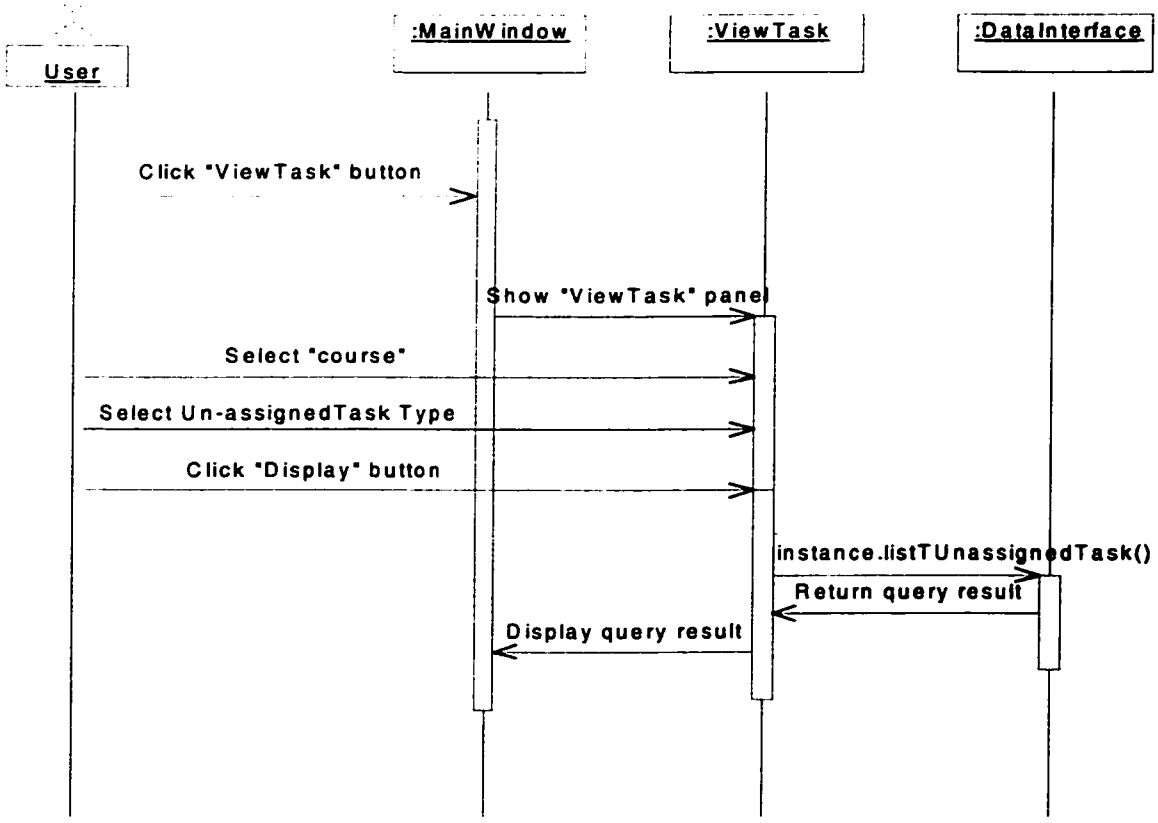

Figure 3-11: System Sequence Diagram: View Unassigned Task

3.4.4 View Assigned Task Use Case

The system sequence diagram for View Assigned Task Use Case is presented in Figure 3-12. The use case starts from user clicking on "ViewTask" button. MainWindow in turn will call (deck.getLayout()).show() to show ViewTask panel. User then selects Course in which TAs registered, select Assigned Task CheckBox. Step by step, user enters display command by clicking on "Display" button, the data which input by user then send to DataInterface by calling listAssignedTask() function. DataInterface will get query results from database and return to ViewTask. The data finally will be displayed in MainWindow for user.

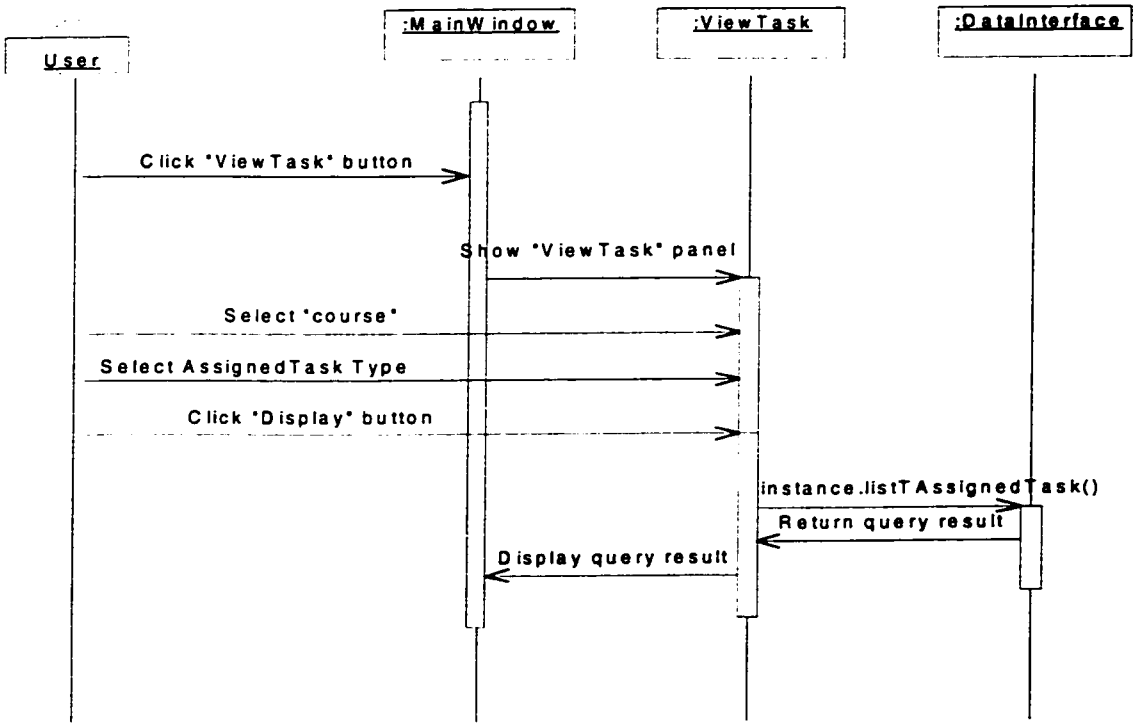

Figure 3-12: System Sequence Diagram: View Assigned Task

3.4.5 View Assignment Use Case

The system sequence diagram for View Assignment Use Case is presented in Figure 3-12. The use case starts from user clicking on "ViewAssignment" button. MainWindow in turn will call (deck.getLayout()).show() to show ViewAssignment panel. User then selects SortBy ComboBox by which TAs are sorted, select Select List ComboBox. Step by step, user enters display command by clicking on "Display" button, the data which input by user then send to DataInterface by calling sortBy() function. DataInterface will get query results from database and return to ViewAssignment. The data finally will be displayed in MainWindow for user.

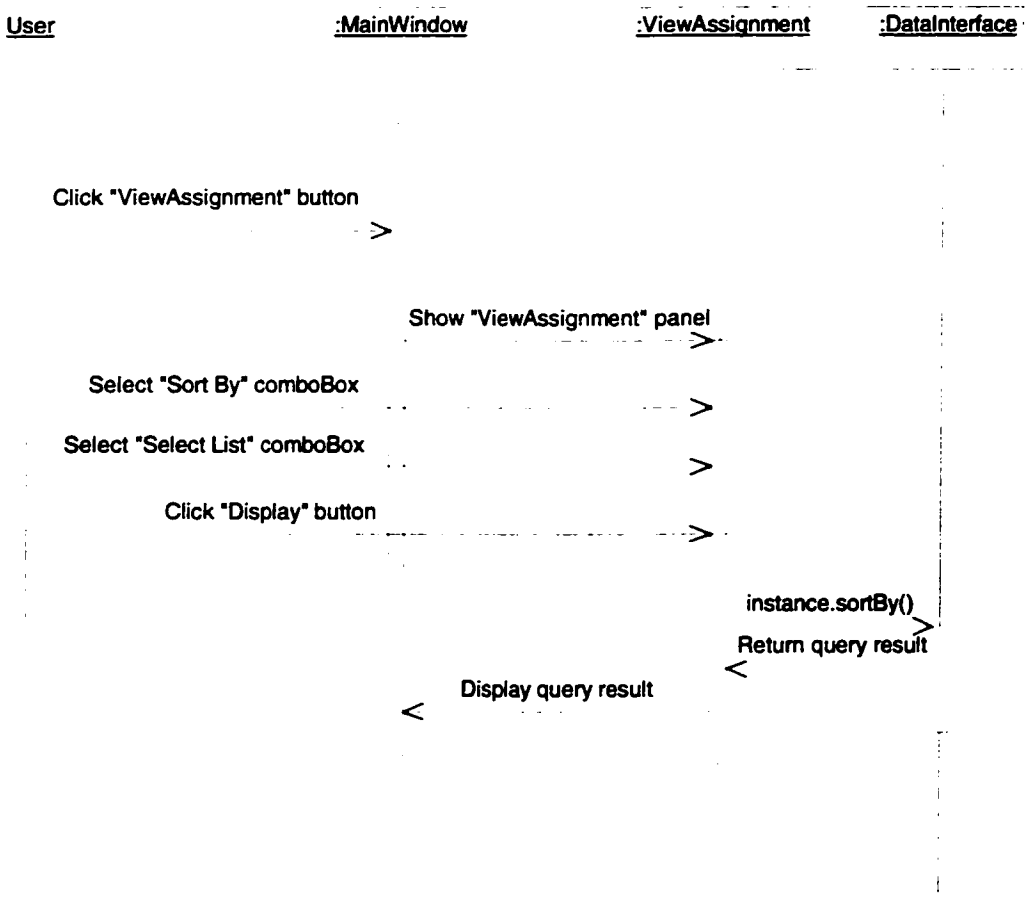

Figure 3-13: System Sequence Diagram: View Assignment

3.4.6 View Generated Assignment Report Use Case

The system sequence diagram for Generate Assignment Report Use Case is presented in Figure 3-14. The use case starts from user clicking on "ViewAssignment" button. MainWindow in turn will call (deck.getLayout()).show() to show ViewAssignment panel. User then select SortBy ComboBox by which TAs are sorted, select "Select List" ComboBox. Step by step, user enters display command by clicking on "Display" button, the data which input by user then send to DataInterface by calling sortBy() function. DataInterface will get query results from database and return to ViewAssignment. The data finally will be displayed in MainWindow for user. At this time, the user gets all the data about assignments. To generate a report for this assignment information, user could enter report command by clicking on "Report" button. Then GReport() is called and a GReport object is created. User can save the data displayed in report window to files.

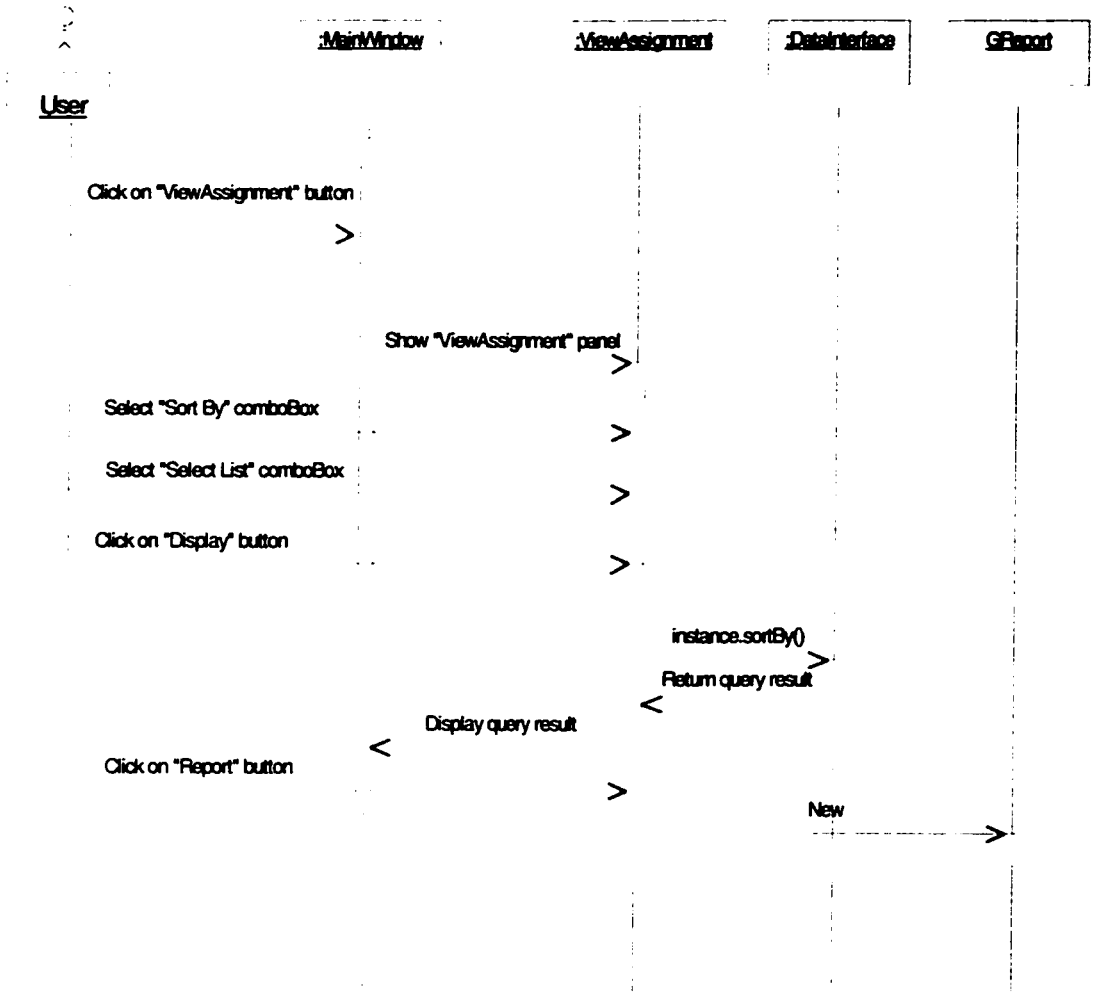

Figure 3-14: System Sequence Diagram: Generate Assignment Report

3.4.7 View Generated TA Report Use Case

The system sequence diagram for Generate TA Report Use Case is presented in Figure 3-15. The use case starts from user clicking on "ViewTA" button. MainWindow in turn will call (deck.getLayout()).show() to show ViewTA panel. User then selects Course in which TAs registered. Step by step, user enters display command by clicking on "Display" button, the data which input by user then send to DataInterface by calling listTA() function. DataInterface will get query result from database and return to ViewTA. The data finally will be displayed in MainWindow for user.

To generate a report about TA information, user could enter report command by clicking on "Report" button. Then GReport() is called and a GReport object is created. User can save the data displayed in report window to files.

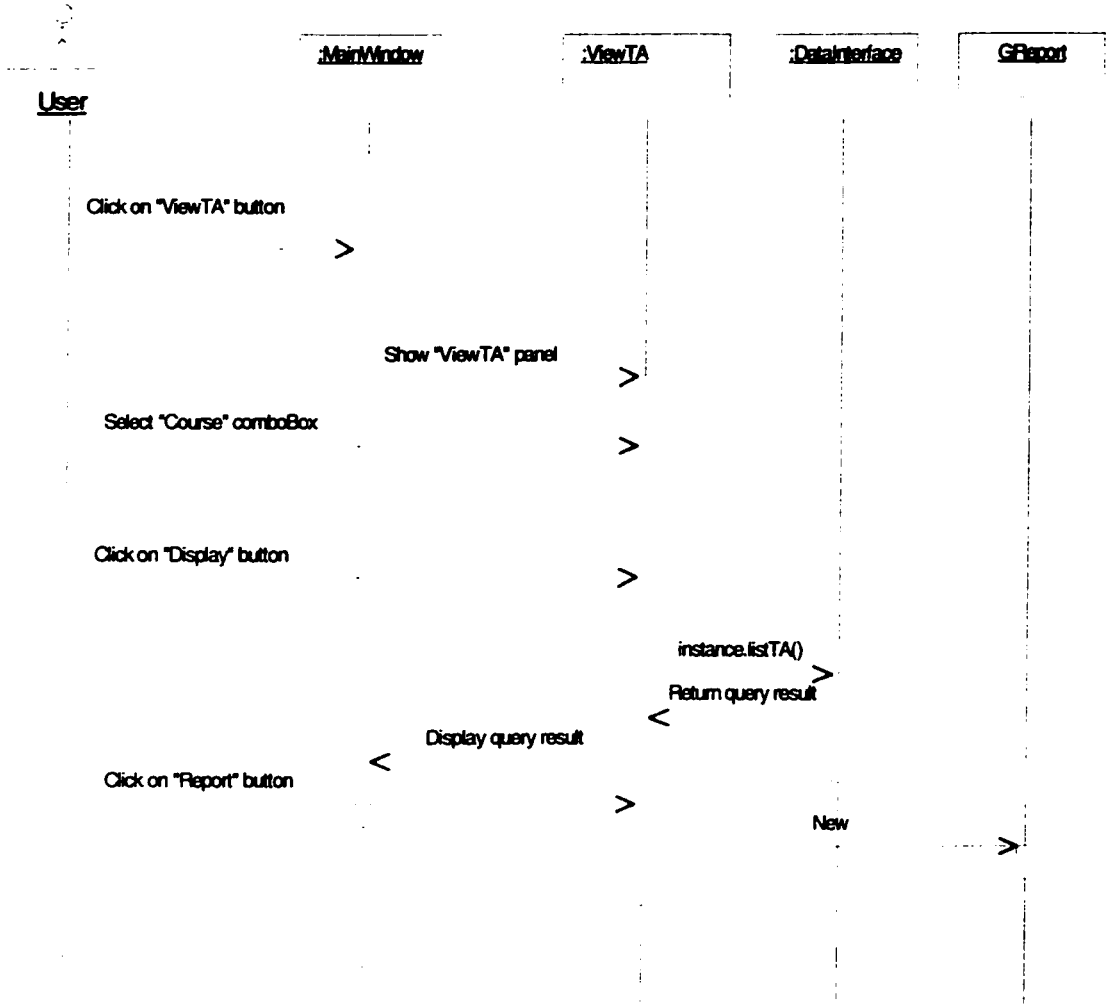

Figure 3-15: System Sequence Diagram: Generate TA Report

3.4.8 View Generated Task Report Use Case

The system sequence diagram for View Task Use Case is presented in Figure 3-16. The use case starts from user clicking on "ViewTask" button. MainWindow in turn will call (deck.getLayout()).show() to show ViewTask panel. User then selects Course in which TAs registered, and select Task Type which TAs need to do. Step by step, user enters display command by clicking on "Display" button, the data which input by user then send to DataInterface by calling listAllTask() function. DataInterface will get query result from database and return to ViewTask. The data finally will be displayed in MainWindow for user.

To generate a report about Task information, user could enter report command by clicking on "Report" button. Then GReport() is called and a GReport object is created. User can save the data displayed in report window to files.

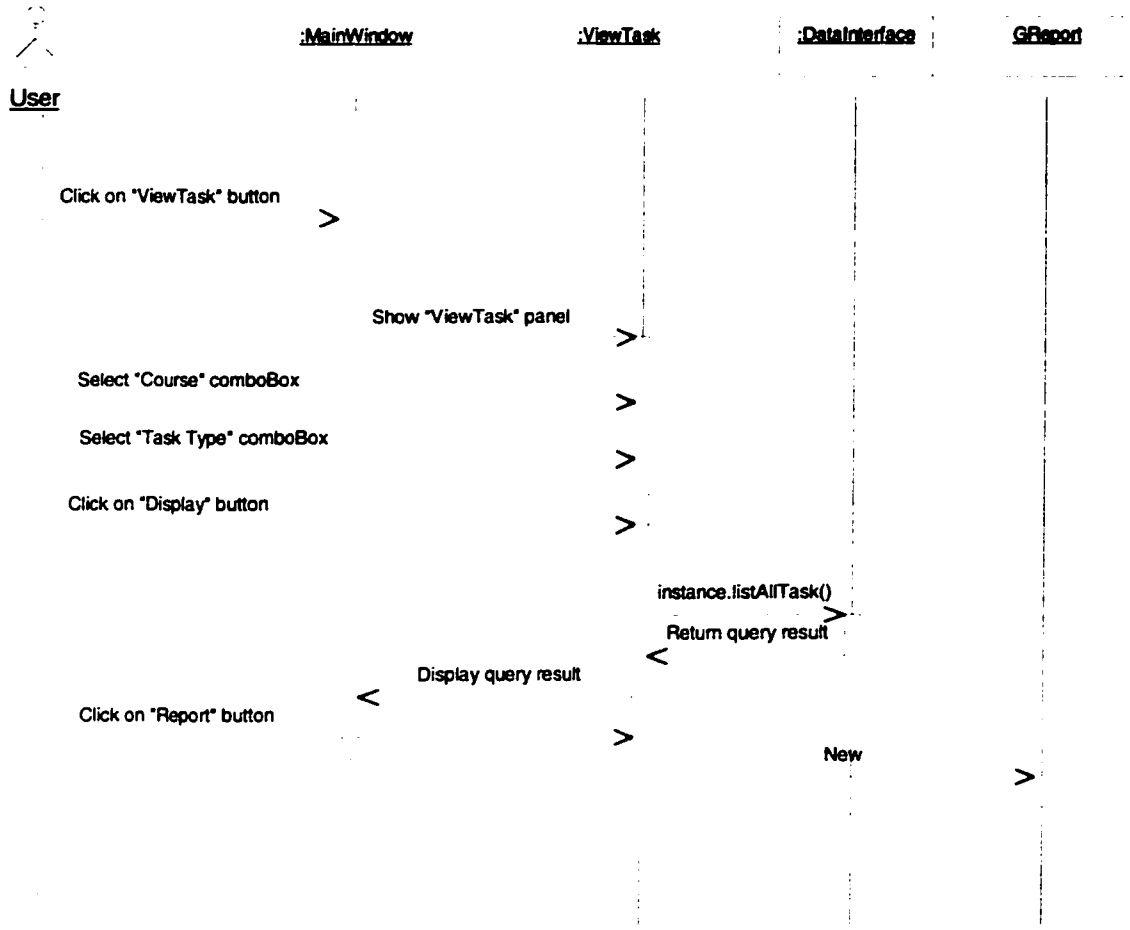

Figure 3-16: System Sequence Diagram: Generate Task Report

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 JBUILDER 5.0 INSTRUCTION

Because the TAAP system is developed in Java using JBuilder 5.0, we introduce some features of JBuilder V5.0.

- JBuilder V5.0 is Borland Software Corporation product. The JBuilder integrated development environment (IDE) provides a single window that is equipped to handle many development functions [4].
- JBuilder also provides many time-saving wizards for application development. With these wizards you can quickly create or modify files, settings, and preferences. The wizards create the framework of your file or application, allowing you to focus on development [4].
- JBuilder provides tools for visually designing and programming Java classes, allowing you to produce new compound or complex components [4].

4.2 JBUILDER 5.0 GENERATED CLASSES IN TAAP SYSTEM

Following classes used in TAAP system are automatically generated by Jbuilder:

- InitialApplicationClass.java

  This class is auto-generated application class for TAAP system. This class consist of a main application class and frame class which can be customized by using the visual designer [4].

- InitialApplicationFrame_AboutBox.java

This class provides TAAP product version, copyright and comments information. You can change information in the About-Box by directly editing the code [4].

4.3 TAAP CLASSES

4.3.1 GUI Module Classes

According to the use cases and task analysis, we choose dialog-based interactive style to achieve the goal of easy-to-use. The dialog-based interactive style provides contextual information for the user, allowing them to make a related set of choices. To support a clear presentation of TA, Task and Assignment information [2], JAVA Swing component called JTable is used in construction of the GUI module. The Button, ComboBox, ProgressBar, Label are also used to give user a clear information about functionality of the TAAP system.

- *MainWindow.java* Class

  This class is the main window GUI class. The TAAP GUI module has been designed as one window system in order to accomplish the goal of simplicity. There are four function buttons on top of the window to display ViewTA, ViewTask, ViewAssignment and AssignTA panel when the corresponding button clicked. So it is easy to use. By default, the ViewTask function is selected when initialize the TAAP system.

  MainWindow class is inherited from Jframe, its major Attributes are declared as below:

```
//...
ViewAssignment viewAssignment = new ViewAssignment();
ViewTA viewTA = new TA();
```

```
ViewTask viewTask = new ViewTask();
AssignTA assignTA = new HelloWorldFrame();
//...
```

The ViewTA, ViewTask, ViewAssignment and AssignTA class are inherited from JPanel, so these sub-GUI part could be put into MainWindow JPanel Container. One of the functionality implemented in MainWindow class is to display ViewTA, ViewTask, ViewAssignment or AssignTA panel on the same location in main window when user click on function button. For this purpose, we implement a layout design method as shown below:

```
JPanel deck = new JPanel();
// ...
deck.setLayout(cardManager);
deck.add(viewTask, "ViewTask");
deck.add(viewTA, "ViewTA");
deck.add(viewAssignment, "ViewAssignment");
deck.add(assignTA, "AssignTA");
CardLayout CardManager2=new CardLayout();
// display viewtask by default
((CardLayout)deck.getLayout()).show(deck,"ViewTask");
//...
```

The above layout design make sure to display different panel in same location. The example of event handler code is given below:

```
void jButton1_actionPerformed(ActionEvent e) {
  // show View TA panel for user in main window
   ((CardLayout)deck.getLayout()).show(deck,"ViewTA");
}
```

The Main Window GUI diagram is presented in Figure 4-1.

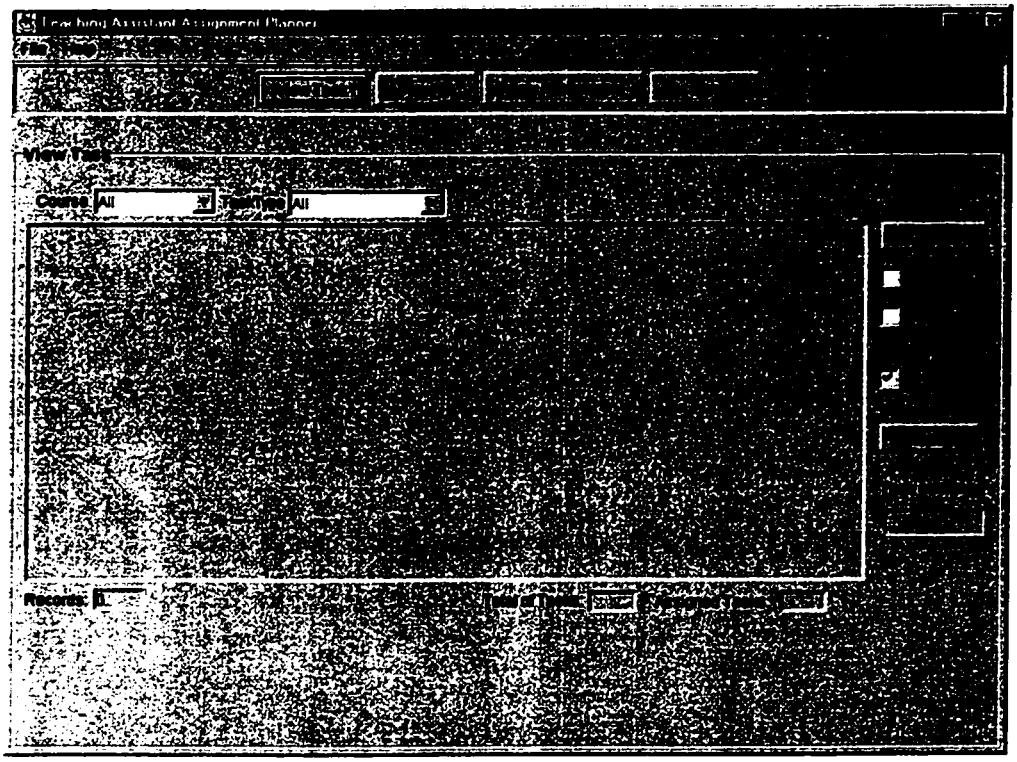

Figure 4-1: The Main Window GUI Diagram

- *ViewTA.java* Class

  This class is designed for View TA functionality. ViewTA class is inherited from Jpanel. In order to display TAs information for different department, we use JcomboBox component for user to choose Course ID. User enters display command by clicking on "Display" button on top of ViewTA panel, the data which input by user then send to DataInterface by calling listTA() function. DataInterface will get

query result from database and return to ViewTA panel. The data finally will be displayed in MainWindow for user.

One of the functionality implemented in ViewTA class is to display a clear presentation of TA information. For this purpose, we implement a design method as shown below:

```
// initial table
 JTable tblTA = new JTable();
 DefaultTableModel tmTask = new DefaultTableModel();
// ...
    void loadTADB()
    {
      tbTAHeader = new Vector();
      String co = (String)cbbCourseSelect.getSelectedItem();
      tbTAHeader.add("Name");
      tbTAHeader.add("SID");
      tbTAHeader.add("AppliedCourse");
      tbTAHeader.add("Task Type");
      tbTAHeader.add("Assign Status");
      tbTAContent = instance.listTA("Tasks","Assignments","TAs",co,jLabel2,jLabel4);
     tmTA.setDataVector(tbTAContent, tbTAHeader);
     tblTA.setModel(tmTA);
      // set column
      TableColumn column;
      TableColumnModel cmod = tblTA.getColumnModel();
      for (int i = 0; i<=4; i++){
      column = cmod.getColumn(i);
      }
    }
```

The View TA GUI diagram is presented in Figure 4-2.

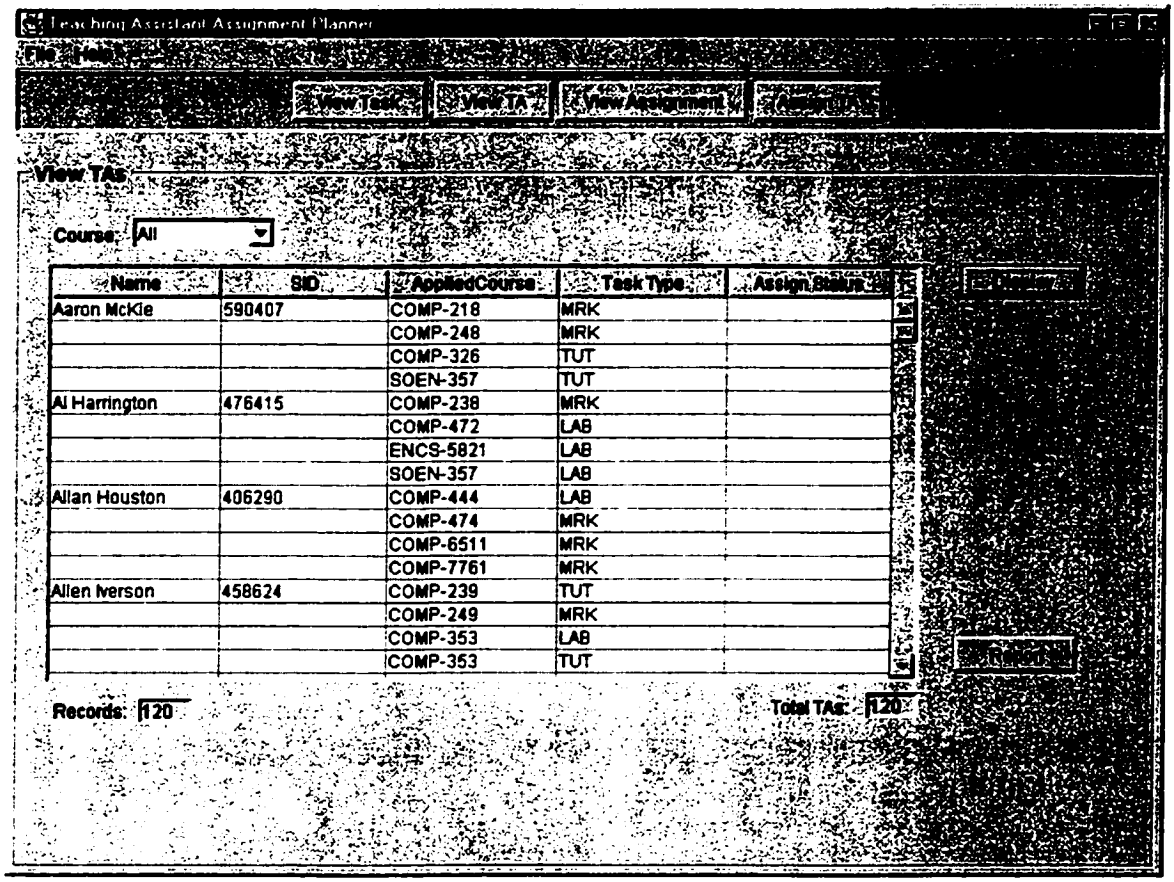

Figure 4-2: The View TA GUI Diagram

One record in Figure 4-2 is equivalent to the group of rows that start with a name and extend to the next name.

- *ViewAssignment.java* Class

  This class is designed for View Assignment functionality. ViewAssignment class is inherited from Jpanel. In order to display TAs information in different order, we use JcomboBox component for user to choose the way of sorting. User enters display command by clicking on "Display" button on top of ViewAssignment panel, the data which input by user then send to DataInterface by calling sortBy() function.

DataInterface will get query result from database and return to ViewAssignment panel. The data finally will be displayed in MainWindow for user.

The functionality for displaying a clear presentation of Assignment information is similar with ViewTA class.

The View Assignment GUI diagram is presented in Figure 4-3. The attribute "ID" in this figure is come from 'Task ID" in Figure 4-4.

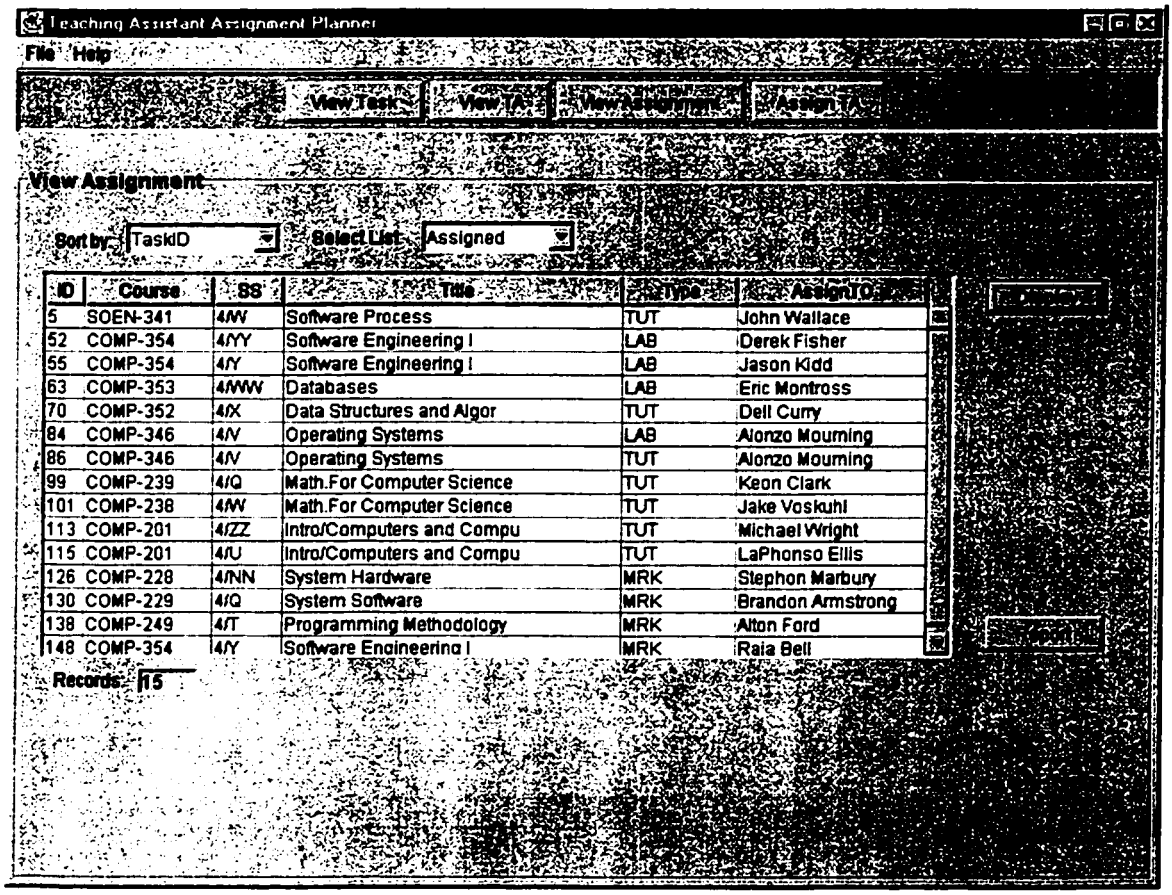

Figure 4-3: The View Assignment GUI Diagram

- *ViewTask.java* Class

This class is designed for View Task functionality. ViewTask class is inherited from Jpanel. In order to display different Tasks information for different department, we

use two JcomboBox components for user to choose Course ID and TaskType. The CheckBox component is also used for user to choose assigned or unassigned task. User enters display command by clicking on "Display" button on top of ViewTask panel, the data which input by user then send to DataInterface by calling listAllTask() or listUnassignedTask() function. DataInterface will get query result from database and return to ViewTask panel. The data finally will be displayed in MainWindow for user.

The View Task GUI diagram is presented in Figure 4-4.

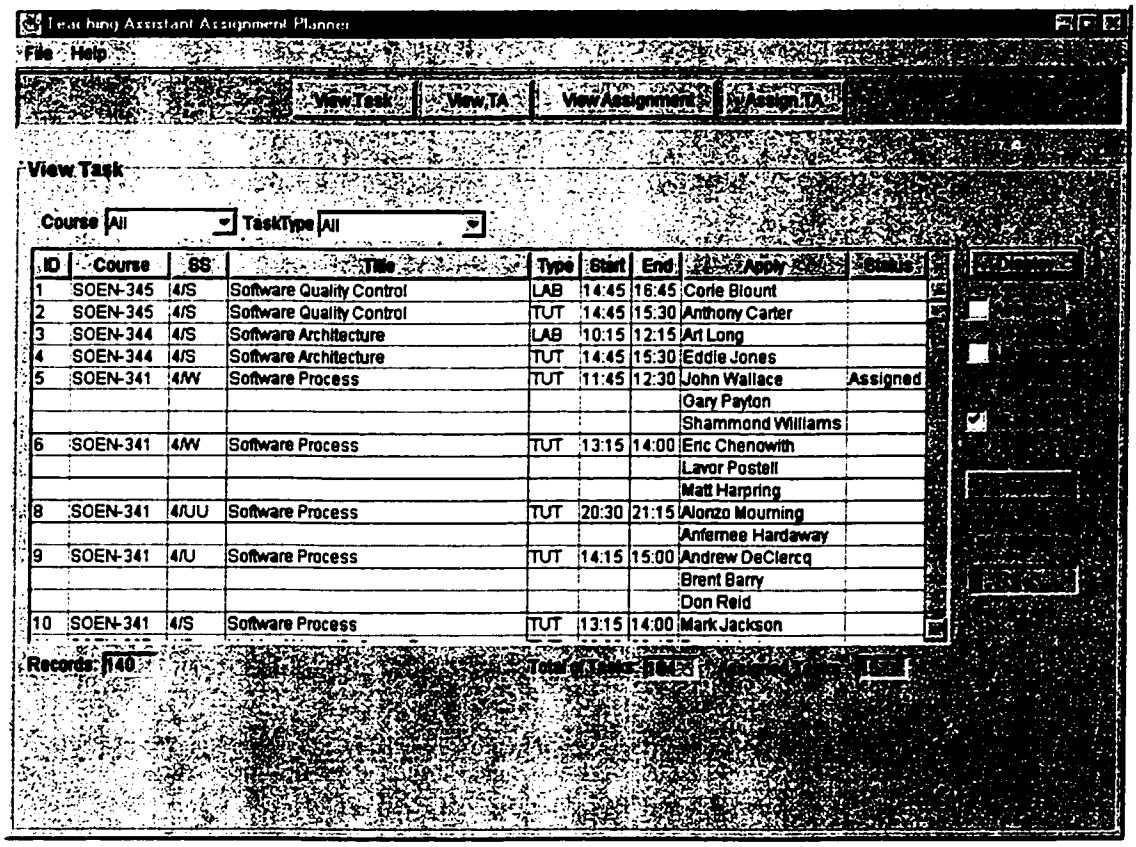

Figure 4-4: The View Task GUI Diagram

- *Greport.java* Class

  This class is designed for Generating Report functionality. GReport class is inherited from Jdialog.

  One of the functionality implemented in GReport class is to display a clear presentation of Report information. For this purpose, we implement a mechanism as shown in APPENDIX A. The mechanism could display report information in Report window with organized format.

  The Generate Report GUI diagram is presented in Figure 4-5.

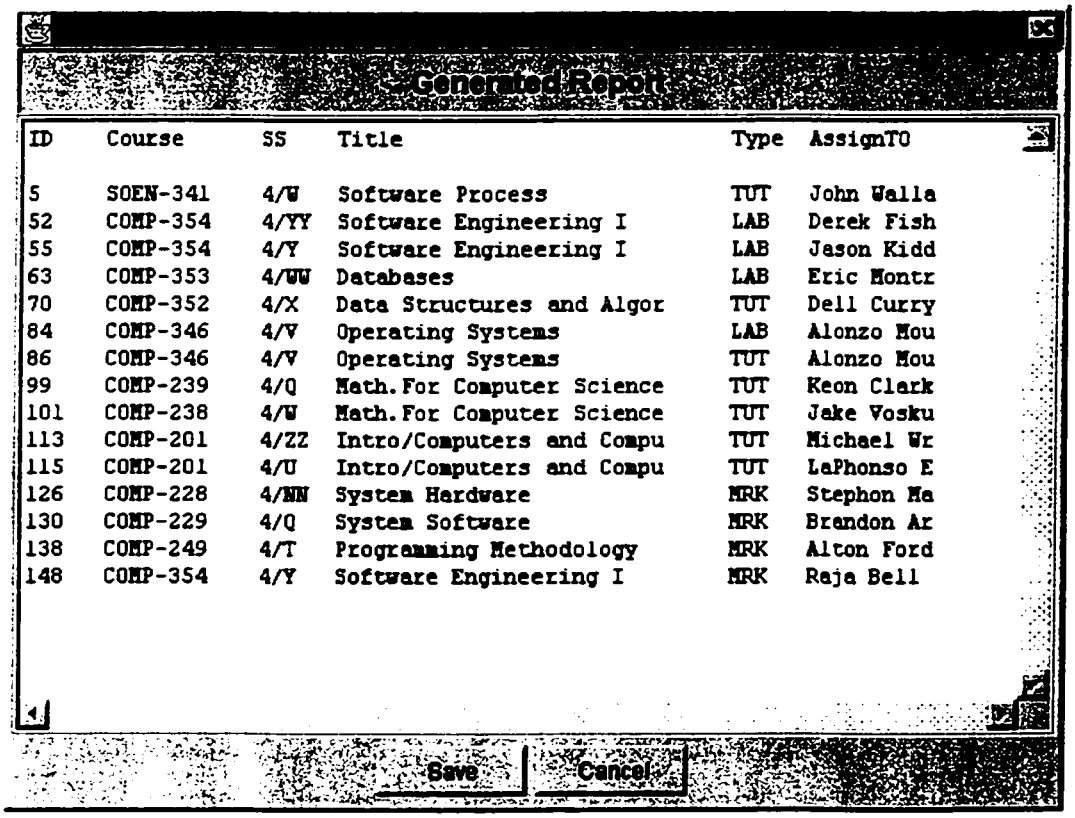

Figure 4-5: The Report GUI Diagram

4.3.2 Data Interface Module Class

- **DataInterface Class**

This class is the interface sending query request to Access database, and get the query results from Access database through JDBC driver. The interface functions will be called be ViewTA, ViewTask, ViewAssignment and AssignTA object.

Methods:

- *public Vector listAllTask(String tableTask, String tableAssign, String tableTA, Object course, Object taskType,boolean assi,JProgressBar pb,JLabel jLabel4,JLabel jLabel6,JLabel jnumber)*

  This method returns a Vector with all tasks

- *public Vector listAssignedTask(String tableTask, String tableTA, String tableAssign, Object course, Object taskType, JProgressBar pb)*

  This method returns a Vector with all assigned tasks.

- *public Vector listTA(String tblTask, String tblAssign, String tblTA, String course,JLabel jl,JLabel j2)*

  This method returns a Vector with all TA information

- *public Vector listUnassignedTask(String tableTask, String tableAssign, Object course, Object taskType)*

  This method returns a Vector with all unassigned tasks

- *public Vector sortBy(String tableTask, String tableAssign,String tableTA,Object sort,boolean assi)*

  This method returns a Vector with all tasks sortby specific order

Attributes:

- *String driverClass = "sun.jdbc.odbc.JdbcOdbcDriver";*

  Indicate the driver information.

- *String url = "jdbc:odbc:SystemDB";*

  Specify the location of the database file.

- *String username = "admin";*

- *String password = "admin";*

  These are username and password for creating connection with Access database.

CHAPTER 5

RESULT

5.1 DEMO

A demo of the TAAP system to Prof. Peter Grogono and Pauline Dubois presents that the TAAP system runs well and meets original requirements. The demo has been done on Lab1022-2 computer room in Computer Science Department in Concordia University. Both Prof. Peter Grogono and Pauline Dubois give us some useful advice for improving the performance of TAAP system.

CHAPTER 6

CONCLUSION

The example on Chapter 5 demonstrates that the implementation of TAAP system runs well and meets the requirements. As we expected, TAAP system could help automate the assignment process in the manner of eliminating human errors, minimizing the memorization request and providing the progressing information and status report. The user could get all the information related to TA, Task and Assignment easily through GUI interface.

6.1 FUTURE WORK

We present the design and implementation of TAAP system with Object-Oriented methodology. One of the limitations in the TAAP system is that we did not use any design pattern in TAAP system. Therefore, we suggest the following further work.

Using MVC architecture in GUI design for TAAP system

The object-oriented MVC model is one of the popular object-oriented software development methods which is particularly useful when the system has a complex and dynamic data structure requiring more than one representation. The model consists of three objects, Model, View, and Control. The data structure is hidden in the Model object, and is accessible only through the View object(s). The Control object serves as a user interface for changing system configuration and parameters of the Model or the View objects. New methods of data visualization can easily be introduced to the system by developing new View objects. [3]

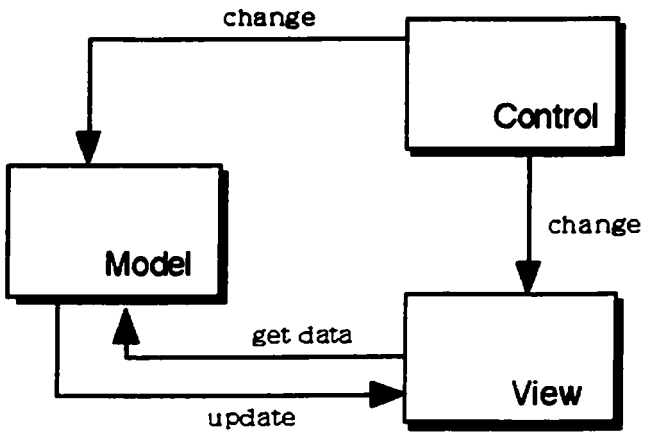

Figure 6-1: MVC architecture

MVC advantages

- Multiple views

  The application can display the state of the model in a variety of ways, and create/design them in a scalable, modular way.

- Ease of updating: controllers and views can grow as the model grows; and older versions of the views and controllers can still be used as long as a common interface is maintained.

- Powerful user interfaces: using the model's API, the user interface can combine the method calls when presenting commands to the user.

- Clarity of designing: the public methods in the model stand as an API for all the commands available to manipulate its data and state. [3]

Bibliography

[1] The Unified Modeling Language User Guide
Addison-Wesley Pub Co, 1998

[2] Xiaomei Yang
Web Based CINDI System GUI Design and Implementation
Master Major Report of Computer Science
Concordia University, 2001

[3] Human-Computer Interaction
Second Edition
Alan J. Dix
Janet E. Finlay

[4] Quick Start
Jbuilder Version 5
Borland Software Corporation

[5] Install Instructions
Jbuilder Version 5
Borland Software Corporation

APPENDIX A - Code of Display Report

```
void listReport(Vector vh,int vhl,Vector vt,int vtl,int tittle)
 {
  String head,content="\n",str_h="";
  StringBuffer text=new StringBuffer(1000);
  int tittle_length=30;
  tittle--;   // add hearder
  try {
  for (int i=0;i<vhl;i++)
  {
    if ((i==0)||(i==2)||(i==4)||(i==6))
    {
              head=(String)vh.get(i);
              text.append(head);
              for (int j=0;j<6-head.length();j++)
                        text.append(" ");
    }
    else if (i==tittle) {
     head=(String)vh.get(i);
    text.append(head);
     if (head.length()<30)
     for (int j=0;j<30-head.length();j++) {
              text.append(" ");
      }
   }
    else {
              head=(String)vh.get(i);
              text.append(head);
              if (head.length()<12)
                        for (int j=0;j<12-head.length();j++) {
                                  text.append(" ");
                        }
              }
    }
text.append("\n\n");
```

```
Vector content0 ;
for (int i=0;i<vtl;i++)
{
 content0 =(Vector) vt.get(i);

 for (int j=0;j<vhl;j++)
 {
  String con=(String)content0.get(j);
  // this.jTextArea1.append(con);
  if ((j==0)||(j==2)||(j==4)||(j==6))
  {
   //  this.jTextArea1.append(con);
    text.append(con);
     for (int k=0;k<6-con.length();k++) {
     //this.jTextArea1.append(" ");
          text.append(" ");
          }
 }
 else if (j==tittle)
 {
  //this.jTextArea1.append(con);
text.append(con);
  for (int k=0;k<30-con.length();k++) {
  //this.jTextArea1.append(" ");
       text.append(" ");
  }
 }
 else
 {
  if (!(con.length()==0))
    if (con.length()<12)
    {
     //  this.jTextArea1.append(con);
     text.append(con);
     for (int k=0;k<12-con.length();k++) {
      text.append(" "); }
  }
```

```
        else if (!(j==tittle)){
                text.append(con.substring(0,10));
                text.append("  ");
        }
       }
      }
      text.append("\n");
      }
      this.jTextArea1.append(text.toString());
  }
catch(Exception ex) {
}
```